1. Write a shell script to display the list of users as well as the number of users connected to the system.

```
=> #!/bin/bash
        echo "Enter LOGNAME or UID"
        read input
        if [[i$input]]&7[$input -eq $input 2>/dev/null]
        then
            echo "Number of terminals are:"
       cat / etc / passwd | grep $input -c
       else
            cat / etc / passwd > userlist
       echo "Number of terminals are:"
       grep -c $input userlist
   fi
```

2. write a shell script that display a listof all files in the current directory to which you have read, write and execute permissions.

```
=> #!/bin/bash
          for var in *
          do
          if test -r$var -a -w$var -a -x$var -a! -d$var
          then
          ls $var
          fi
          done
```

3. To list the files according to modification or access time depending on the argument in command line

```
=> case $1 in
```

```
  lm) ls -lt;;
  la) ls -lut;;
 *) ls -l;;
 esac
```

4. Write a shell script to display the files created or updated within fourteen days from the current date.

=> find -atime -14 -mtime -14 | sort -u

5. Develop a shell script which display all files with all attributes those have been created or modifies in the month of november.

```
=>for var in *
  do
   set --`ls -l $var `if test "$6"="Nov"
  then
   ls -l$var
  fi
```

6. Write a shell script which reports names and size of all files in a directory whose size exceeds 100 bytes. the file names should be printed in decreasing order of their size. The total number of such files should be reported.

```
=>#!/bin/bash
  if test $# -ne1
  then
    echo "Please giva a directory name and try again:"
  exit
 fi
 cd $1 find -size 100b | sort -nr
 echo "Total number of such files:"
 find -size +100b | grep -c ".*"
```

7. Write a shell script that shows the names of all the non-directory in the current directory and calculates the sum of size of them.

```
=> dir | awk `{total+=$4}
        END (print total)
        ls -l;;
```

8.Write a shell script to list the names of files under the current directory that starts with a vowel.

```
=> ls |grep "^[aeiou]"
```

9. Write a shell script which receives tw filename as argument and check whether the two file`s contents are same or not. If they are same then the second file should be deleted.

```
=>if test $# -ne2
    then
        echo "Please give the two filenames:"
    exit
    fi
    cmp -s $1 $2
    if test $? -eq0
    then
      echo "$1 and $2 are same"
    rm $2
    else
        echo "$1 and $2 are not same"
    fi
```

10. A file called list consist of several words. Write a shell script which will receive a list of filenames, the first of which would e list. The shell script should report all occurrences of each word in list in the rest of the files supplied as arguments.

```
=>if [$# -eq0]
    then
```

```
echo "no argument"
else
tr " " "
"<$1> temp
shift
for i in $*
do
tr " " "
"<$i> temp1
y= wc -l <temp`
j=1
while [$j -le $y]
do
x=`had -n $j temp | tail -1`
c=`grep -c "$x" temp1`
echo `expre $j 1`
done
done
fi
```

11. Write a shell script which delets all lines containing the word UNIX in the files supplied as arguments to the sell script.

```
=>if [$# -lt 1]
then
echo "Check the argument once"
exit
fi
```

```
echo "Enter a word"

ead word

for file in $*

do

grep -iv "$word" $file | tee 1>/dev/null

done

echo "Lines containing given word are deleted"
```

12. Write a shell script which get executed the moment users login. It should display the message 'GOOD MORNING', 'GOOD AFTERNOON' or 'GOOD EVENING' depending upon the time at which users log in.

```
=>h=$(date +"%H")

      if [ $h -gt 6 -a $h -le 12 ]

      then

      echo Good Morning

      elif [ $h -gt 12 -a $h -le 16 ]

      then

      echo Good Afternoon

      elif [ $h -gt 16 -a $h -le 20 ]

    then

    echo Good Evening

    else

    echo Good Night

    fi
```

13. **1. Search a file with specific name.**
$ find ./GFG -name sample.txt It will search for sample.txt in
GFG directory.

2. Search a file with pattern.
$ find ./GFG -name *.txt It will give all files which have '.txt' at the end.

3. Search for empty files and directories.
$ find ./GFG -emptyThis command find all empty folders and files in the entered directory or sub-directories.

4. Search for file with entered permissions.
$ find ./GFG -perm 664

14. set assigns it's argument to the positional parameters

Example :

```
#!/bin/bash

set   `date`

Echo " the year of system = $6"
Echo " the month of system = $4"
```

15.Write a shell script which displays the message "welcome" and

prints the date when you log in to your system.

```
=>h=$(date +"%H")

      if [ $h -gt 6 -a $h -le 12 ]

      then

      echo Welcome

      elif [ $h -gt 12 -a $h -le 16 ]

      then

      echo Welcome

    elif [ $h -gt 16 -a $h -le 20 ]

    then

    echo Welcome

    else

    echo Welcome

    fi
```

16.Write a shell script to check if a given file (filename supplied as command line argument) is a regular file or not and find the total number of words, characters and lines in it

=>echo

```
c=$( wc -c < test.txt)
echo "Number of characters in test.txt is $c"
echo
w=$( wc -w < test.txt)
echo "Number of words in test.txt is $w"
echo
l=$( ec -l < test.txt)
echo "Number of lines in test.txt is $l"
```

17. Write a shell script to check whether the given file is a blank file or not. If not found blank then display the contents of the file.

=>#!/bin/sh

```
f='afile.txt'

hasData ()
{
    echo "$f has data."
    ls -l $f
    cat $f
}

noData()
{
    echo "$f is empty."
```

```
    ls -l $f
}

if [[ -s $f ]]
then
    hasData
else
    noData
fi
```

18.Write a shell function size( ) which lists only the total size of the files(filenames supplied as arguments).

```
=>#!/bin/bash
FILENAME=/home/heiko/dummy/packages.txt
FILESIZE=$(stat -c%s "$FILENAME")
echo "Size of $FILENAME = $FILESIZE bytes."
```

19. Write a shell script to make a password based menu-driven program, which will give a maximum of three chances to enter the password. If the g p pg iven password is correct then the program will show the □ Number of users currently logged in. □ Calendar of current month. □ Date in the format: dd / mm / yyyy. □ Q i u t The menu should be placed approximately in the centre of the screen

```
=>read -s -p "Enter Password: "  pswd
echo -e "\nYour password is: " $pswd

echo "SELECT YOUR FAVORITE OPTION";
echo "1. UERS CURRENTLY LOGIN"
echo "2. CALENDER"
echo "3. DATE IN FORMAT:DD/MM/YYYY"
echo "4. Exit from menu "
```

```
echo -n "Enter your menu choice [1-4]"
while :
do
read choice
case $choice in
   1)   echo "You have selected the option 1"
                echo "$(w)";;
   2)   echo "You have selected the option 2"
                echo "$(cal)";;
   3)   echo "You have selected the option 3"
                echo "$(date +%d/%m/%y)";;
   4)   echo "Quitting ..."
            exit;;
   *) echo "invalid option";;

 esac
 echo -n "Enter your menu choice [1-4]: "
 done
```

20.Sort the file/etc/psswd on GUID(primary) and UID(secondary) so that theusers with the same GUID are placed together. Users with a lower UID should be placed higher in the list.

```
=> echo "$(cut -d ":" -f 3,4/etc/passwd | sort -n)"
```